The Programmer Writing

Prospecting for programmer's gold.

IN 1987, FRED BROOKS OBSERVED THAT "the gap between the best software engineering practice and the average practice is very wide—perhaps wider than in any other engineering discipline. A tool that disseminates good practice would be important" ("No Silver Bullets: Essence and Accidents of Software Engineering," *Computer*, April 1987). Continuing Brooks' line of thought in 1990, the US National Research Council's Computer Science and Technology Board stated that the biggest gains in software development quality and productivity will come from disseminating effective practices—codifying, unifying, and distributing existing knowledge via software engineering handbooks ("Scaling Up: A Research Agenda for Software Engineering," *Communications of the ACM*, March 1990).

Who should write these handbooks?

In August 1837, Ralph Waldo Emerson delivered an address that came to be known as "The American Scholar" (*Selections from Ralph Waldo Emerson*, Stephen E. Whicher, ed., Houghton Mifflin, 1960). It is one of the most inspirational essays I have read; moreover, I believe it contains the answer to the question of who should write these software handbooks.

EMERSON'S AUTHORS. Most software development books are written by six kinds of authors: recent retirees, university professors, seminar instructors, consultants, think-tank developers, and developers working on production software. People in each of these groups have valuable contributions to make, and it would be a mistake to discount any of them. Recent retirees bring years of experience, insight, and reflection to their writing. University professors bring a full awareness of leading-edge research. Seminar instructors have a chance to test their material in front of hundreds or even thousands of students before publishing their material in book form. Consultants see dozens of clients a year, and their observations can be based on an incredible breadth of exposure to effective and ineffective software practices. Think-tank developers at Xerox PARC, AT&T Bell Labs, and similar environments have produced some of our best software engineering technology. But I think that developers working on production software must shoulder the primary burden of creating these handbooks.

In "The American Scholar," Emerson draws a distinction between a thinker and Man Thinking

The "Man Thinking" has a strong bias toward action: direct experience is critical to genius.

(which is his synonym for American Scholar). A thinker is someone whose sole function is to think. A thinker experiences life secondhand, through other people's books, articles, and descriptions of the active world. A Man Thinking, on the other hand, is a robust person who is active in the world, actively engaged in a trade or occupation, who occasionally pauses for reflection. The Man Thinking has a strong bias toward action. "The true scholar grudges every opportunity of action past by as a loss of power. It is the raw material out of which the intellect molds her splendid products." Emerson argues that the direct experience of Man Thinking is critical to genius and that genius can emanate only from a Man Thinking, not from a mere thinker.

Emerson says that immersion in the active world is essential to understanding what other Men Thinking have written about it. Readers who do not bring a base of action to their reading will understand little of what they read. But, "When the mind is braced by labor and invention, the page of whatever book we read becomes luminous with manifold allusion. Every sentence is doubly significant and the sense of our author is as broad as the world."

If readers not "braced by labor and invention" get little out of what they read, writers not braced by labor and invention can put little into what they write. As Emerson says, action is essential. Without

Continued on page 119

Editor:
Steve McConnell
Construx Software Builders
PO Box 6922
Bellevue, WA 98008
stevemcc@construx.com

Continued from page 120

it, thought can never ripen into truth. Readers instantly know whose words are loaded with life, and whose are not. "I learn immediately from any speaker how much he has already lived, through the poverty or the splendor of his speech. Life lies behind us as the quarry from whence we get tiles and copestones for the masonry of today. Colleges and books only copy the language which the field and the work-yard made."

MISSING THE BOAT. Thinkers who cannot write authentically have what I call the James Fenimore Cooper syndrome. Cooper was an early American writer who was fascinated by the American Indians and wrote several stories about them, including *The Deerslayer* and *The Last of the Mohicans*. Cooper's writing was later skewered by the great American humorist Mark Twain as being hopelessly inaccurate fantasy ("Fenimore Cooper's Literary Offenses" [1895] in *How to Tell a Story and Other Essays*, Oxford Univ. Press, 1996).

In a scene from *The Deerslayer*, Cooper has six Indians climb onto a sapling overhanging a river to wait for a scow being hauled upstream by rope. The Indians' plan is to jump onto the roof of a cabin on the scow, which is 90 feet long and 16 feet wide at its widest point. The six Indians try to time their jumps onto the cabin roof, but one falls short of the roof and lands on the boat's stern; the remaining five miss the boat entirely and land in the water.

Twain had been a riverboat pilot, and he took Cooper to task for his sloppy description of a subject Twain knew intimately. Twain pointed out the implausibility of a sapling supporting the weight of six adult men. He then pointed out that according to Cooper's description of the river, a boat one-third the length of Cooper's would have had difficulty navigating the twists and turns in the river.

Developers will not buy books that, like Cooper's Indians, miss the boat.

Cooper's scow would have wedged itself into a corner halfway through the first turn. As for the Indians' jumping, the maximum speed the scow could be pulled upstream was about one mile per hour, which would have given the Indians a minute and a half to jump onto the boat and a full minute to jump onto the cabin roof. That amount of time hardly calls for precise timing, but Cooper's Indians managed to miss the cabin anyway. Perhaps they lost their concentration when they realized that Cooper's river at that point was only two feet wider than the boat; they could have saved themselves some trouble by simply stepping onto the boat as it scraped past them. But, as Twain says, because Cooper didn't allow his Indians to hop on from the shore, their mishap is his fault, not theirs.

At the 17th International Conference on Software Engineering, David Parnas pointed out that papers from earlier conferences that had received awards for being "most influential" had arguably not been influential at all ("On ICSE's Most 'Influential' Papers," *Software Engineering Notes*, July 1995). I think the James Fenimore Cooper syndrome is part of the reason. These papers might influence researchers, but they do not influence practitioners because, to practitioners, the methodologies they describe seem about as authentic as Cooper's stories.

AUTHENTICITY. Practicing software developers are every bit as skeptical about software development handbooks as Twain was of Cooper's prose. Software methodology books are seen as theoretical, applicable only to small projects, hard to adapt, inefficient, and incomplete. Software authors sometimes bemoan the fact that the average software developer buys less than one software development book a year, but the reason for that is not such a great mystery. Developers will buy books that have been "braced by labor and invention," but not books that, like Cooper's Indians, miss the boat.

Mark Twain argued that the best frontier adventure stories were written by people with keen eyes for detail who had actually lived on the frontier, and I am convinced that the best software handbooks will be based on the work of software developers who have recently lived through production software projects. To apply Emerson's point to software engineering, the tiles and copestones of software engineering handbooks must come from The Programmer Writing, from people who are actively participating on production software projects and who occasionally take time to reflect on their work and write about it.

Practicing software developers are every bit as skeptical about handbooks as Twain was of Cooper's prose.

A CALL TO ACTION. If you are actively developing software, I urge you to write about your insights. If you have worked on a project that taught you valuable lessons, share them—whether they are coding details, quality assurance practices, more effective project management, or even a software development topic that doesn't have a name yet. Submit your writing to *IEEE Software* or some other magazine, or fully develop your ideas into a book. If your insights are stronger than your writing, find a consultant, seminar instructor, university professor, or other skilled writer to be your co-author. You don't need to worry that what you have learned won't apply to other people's projects. As Emerson says, "Success treads on every right step. For the instinct is sure, that prompts him to tell his brother what he thinks. He then learns that in going down into the secrets of his own mind he has descended into the secrets of all minds. The deeper he dives into his privatest secretest presentiment, to his wonder he finds this is the most acceptable, most public, and universally true." ◆